

Live Inspection of Spreadsheets

Daniel Kulesz*, Fabian Toth† and Fabian Beck‡

University of Stuttgart

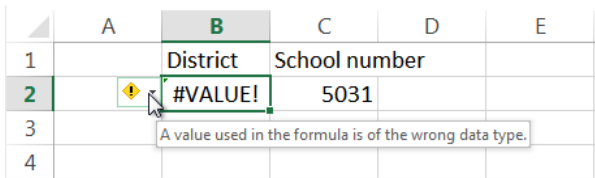
*daniel.kulesz@informatik.uni-stuttgart.de, †tothfn@studi.informatik.uni-stuttgart.de, ‡fabian.beck@visus.uni-stuttgart.de

Abstract—Existing approaches for detecting anomalies in spreadsheets can help to discover faults but they are often applied too late in the spreadsheet lifecycle. By contrast, our approach detects anomalies immediately whenever users change their spreadsheets. This live inspection approach has been implemented as part of the Spreadsheet Inspection Framework, enabling the tool to visually report findings without disturbing the users’ workflow. An advanced list representation allows users to keep track of the latest findings, prioritize open problems, and check progress on solving the issues. Results from a first user study indicate that users find the approach useful.

I. INTRODUCTION

Faults in spreadsheets are common and can cause severe damage [1]. In recent years, several tool-based approaches have been developed for automatically detecting anomalies in spreadsheets [2], arguing that anomalies are dependable indicators for possible faults in spreadsheets. Today, a number of anomaly detection tools aims for a tight integration into spreadsheet environments like Microsoft Excel. One key benefit of tightly integrated tools is their ability to communicate findings in the same environment users work with their spreadsheets. Findings vary by detection approach and can be, e.g. smelly formulas, failed test cases, or violated constraints.

Existing spreadsheet anomaly detection tools still suffer from a major drawback: Since their scans have to be triggered manually, it cannot be guaranteed that users execute them regularly—if at all. The more actions users take and the more time passes between scans, the more difficult it gets for users to identify the actions responsible for the reported findings. Also, the spreadsheet could contain findings already created by previous users, and distinguishing between new and old findings puts an additional mental load onto users.



	A	B	C	D	E
1		District	School number		
2		#VALUE!	5031		
3					
4					

Fig. 1: Anomaly reported by Microsoft Excel 2013

At first glance, checking spreadsheets for anomalies automatically in the background seems to be the trivial solution to address the described issues. In fact, recent versions of Microsoft Excel already issue live inspection techniques by providing warning icons for built-in error types and report them as warning diamonds next to the affected cells (Figure 1). However, adopting a similar approach for more complex

anomalies involves several challenges. In the rest of this position paper, we describe these challenges, our attempts to overcome them, and first experiences with the resulting solution.

II. LIVE INSPECTION CHALLENGES

We identified the following challenges for a live inspection approach of spreadsheets:

- Ch1: Avoid disruption: Users primarily want to work with their spreadsheet and not study findings. Thus, users must be notified in a low-disruptive manner.
- Ch2: Support workflow: Users should be free to choose when to deal with findings and supported in their workflow addressing the findings.
- Ch3: Motivation: Users should be motivated to deal with reported findings.
- Ch4: Recent first: Findings caused by recent user actions should be reported in a more prominent fashion than older findings.
- Ch5: Provide overview: Users need an overview of all currently open findings and support to group them.

III. LIVE INSPECTION APPROACH

In previous work [3], we developed an open-source tool named Spreadsheet Inspection Framework (SIF)¹. It allows users to scan spreadsheets for anomalies using a number of automated and partly automated detection techniques. SIF visualizes findings in a side pane and with in-spreadsheet marker icons (example in Figure 2, cell C12). We extend this tool by a live inspection mechanism.

The live inspection mechanism borrows ideas from the task metaphor that many e-mail clients employ to empower users to process and categorize incoming e-mails [4]. Users typically do not sit and wait for new e-mails to arrive but do other work. Similarly, spreadsheet users are trying to solve a task with their spreadsheet when being informed about a new finding they caused. Even if the findings are a direct cause of the users’ actions, the situation may be comparable in terms of the amount of attention users are willing to invest.

To adopt the metaphor, we divided the formerly flat list (Ch5) of findings (anomalies) into four categories represented by the tabs shown in Figure 2. They are named, from left to right: ‘Open’, ‘Later’ (postponed), ‘Ignored’ and ‘Archive’ (solved). Figure 3 illustrates the states between which findings can travel.

¹<http://spreadsheet-inspection-framework.github.io>

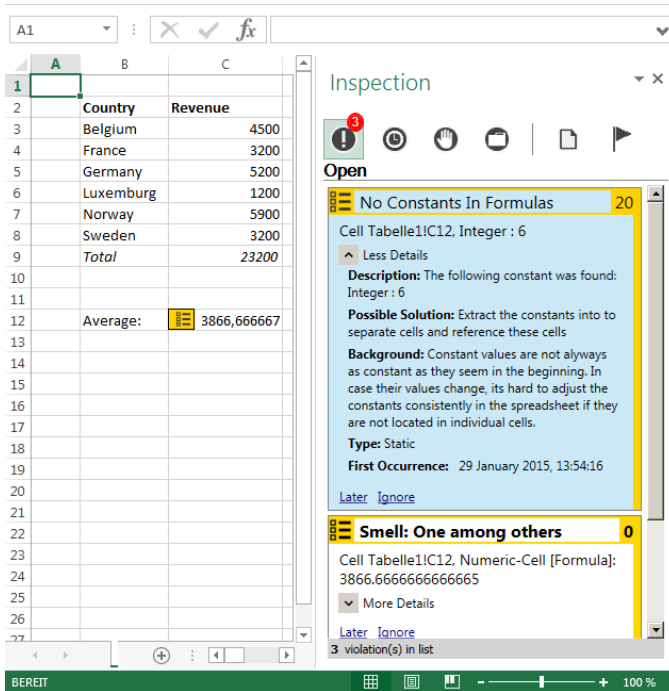


Fig. 2: Extended inspection pane of SIF

A newly detected finding is added to the list of findings as 'Open'. Open findings have a state of being 'read' or 'unread'. The eye-catching red bubble next to the 'Open' section shows the number of unread findings (Ch1 Ch3, Ch5). Recent findings are placed on top (Ch4). The tabs 'Later' and 'Ignore' do not employ this notification mechanism because findings become part of these categories only when users explicitly move them there (Ch2)—by definition, all findings of these categories are read.

Findings that were raised but are meanwhile solved move to the 'Archive' without triggering a notification. The archive is a neutral category. In a preliminary version we experimented with a reward mechanism that highlighted solved findings and counted them using a green notification bubble (Ch3), but since many findings can be 'solved' by simply deleting the causative cells, this counter could introduce false incentives.

On the technical side, we implemented a diff mechanism which allows us to distinguish new from existing findings. Additionally, we set a trigger that automatically issues a scan whenever a user's action triggers a recalculation of the spreadsheet. An extended preferences dialog allows users to choose which inspection rules shall be included in the automatic scan on a per-spreadsheet basis.

IV. EVALUATION

We did a first user study with one pilot followed by five participants (male engineering students aged between 18 and 24). The participants had to solve two tasks in a given spreadsheet that required extending and changing the business logic of its formulas. The spreadsheet was designed in such a way that findings were likely to be caused.

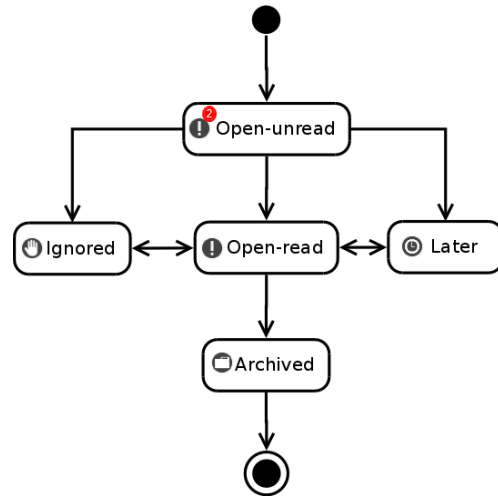


Fig. 3: State diagram for finding states

Three of the participants used the in-spreadsheet marker icons and directly tried to solve open issues as they appeared, but skipped findings they did not find trivial in the first place. They kept the inspection pane closed and did not open it until they finished their primary tasks. Then, they used the inspection pane to solve the remaining findings. The fourth participant did not pay attention at all to the findings until he finished the primary task. In contrast, the fifth participant kept the inspection pane open all the time and paid more attention to the findings than to the actual task.

The first three subjects achieved a rapid learning effect: Once they understood that constants in formulas lead to findings, they changed their behavior and did not put constants into the next formulas they created. Overall, the participants rated the live inspection mechanism to be low distracting and generally acceptable.

V. FUTURE WORK

Future work should provide an extended reward mechanism to motivate the users to solve open findings. When processing large spreadsheets or checking for more complex anomalies, users should be able to continue working while the scans are still running. Also, checks need to be executed incrementally and in order of priority, comparable to selecting and prioritizing regression tests in software engineering.

REFERENCES

- [1] S. G. Powell, K. R. Baker, and B. Lawson, "A critical review of the literature on spreadsheet errors," *Decision Support Systems*, vol. 46, no. 1, pp. 128–138, 2008.
- [2] D. Jannach, T. Schmitz, B. Hofer, and F. Wotawa, "Avoiding, finding and fixing spreadsheet errors—a survey of automated approaches for spreadsheet QA," *Journal of Systems and Software*, 2014.
- [3] D. Kulesz, J. Scheurich, and F. Beck, "Integrating anomaly diagnosis techniques into spreadsheet environments," in *Software Visualization (VISSOFT)*, 2014 Second IEEE Working Conference on. IEEE, 2014, pp. 11–19.
- [4] A. M. Szóstek, "Dealing with my emails: Latent user needs in email management," *Computers in Human Behavior*, vol. 27, no. 2, pp. 723–729, 2011.